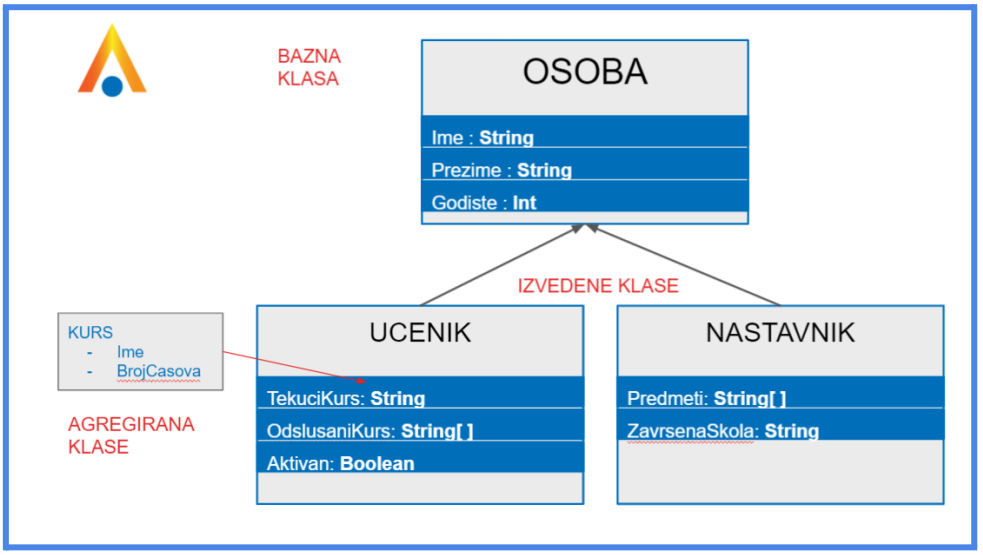
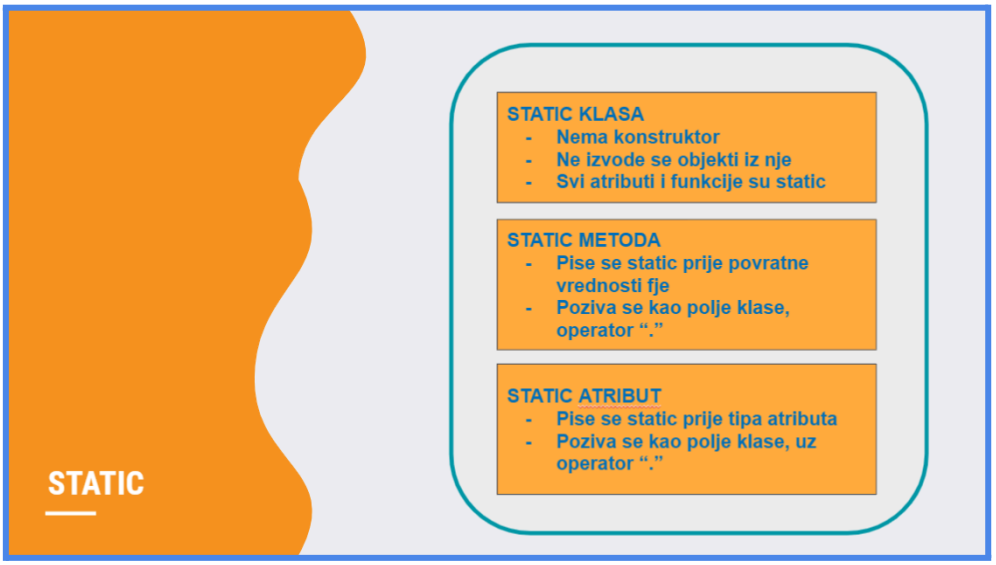
****

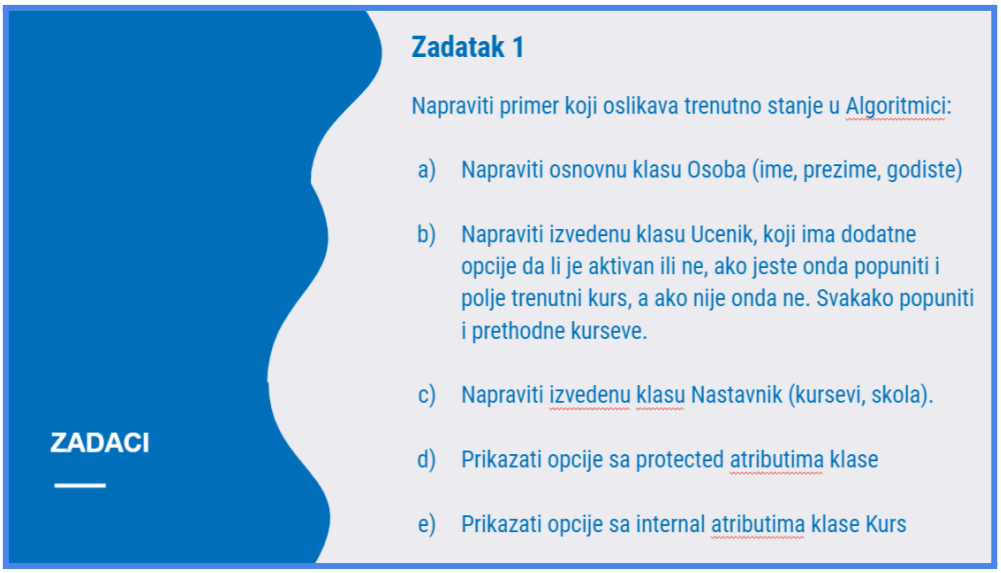
Tema 11

****



- ➔ **Prisetiti se sta smo o ovome radili iz Pythona**
- ➔ **Nasledjivanjem se smanjuje copy paste proces u programiranu jer umesto da se kopira, funkcionalnosti se nasledjuju**
- ➔ **Logicko razmisljanje i kriticki stav - potrebno radi definisanja sta je to sto moze nesto da nasledi**
- ➔ **Pojmovi:**
  - ◆ **Bazna klasa - osnovna klasa, maticna klasa**
  - ◆ **Izvedene klase - klase koje nastaju iz bazne i nasledjuju njene atribute i metode**
  - ◆ **Agregiranja klasa - klasa koja moze da se koristi kao polje u drugoj klasi**
  - ◆ **Staticna klasa - klasa iz koje se ne izvode objekti, koja moze da se tretira kao pomocna klasa**





PRIMER 1 - Osnovni primer za nasledjivane klase.

KORAK 1	OSNOVNA KLASA
KORAK 2	IZVEDENE KLASE
KORAK 3	STATICNE KLASE I METODE ZA TESTIRANJE
KORAK 4	AGREGATNA KLASA

```
using System;
using System.Linq;
using System.Collections;
using System.Collections.Generic;

namespace PrimerNasledjivanje
{
    // staticka klasa - ima sve metode i polja static
    // staticka klasa - nema konstruktore, ne pravi instance
    public static class Program
    {
        public static List<Ucenik> uceniciUAlg = new List<Ucenik>(10);
        public static List<Nastavnik> nastavniciUAlg = new
List<Nastavnik>(2);
        public static List<Kurs> kursevi = new List<Kurs>(5);

        public static void PopuniListuKurseva()
        {
            Kurs Python1 = new Kurs("python1", 14);
            Program.kursevi.Add(Python1);
            Kurs Python2 = new Kurs("python2", 14);
            Program.kursevi.Add(Python2);
            Kurs Pygame1 = new Kurs("pygame1", 14);
            Program.kursevi.Add(Pygame1);
            Kurs Htmlcss1 = new Kurs("htmlcss1", 14);
            Program.kursevi.Add(Htmlcss1);
            Kurs ScratchJR = new Kurs("scratchJR", 12);
            Program.kursevi.Add(ScratchJR);
            Kurs Tynker = new Kurs("tynker", 12);
            Program.kursevi.Add(Tynker);
```

```
        }

        public static void PopuniListuUcenika()
        {
            // kada se doda klasa za kurseve koristiti ovaj zapis
            string[] kurseviAna = new string[] { kursevi[0].ime,
kursevi[2].ime, kursevi[3].ime };
            string[] kurseviAndjela = new string[] { kursevi[0].ime,
kursevi[1].ime, kursevi[2].ime, kursevi[3].ime };
            string[] kurseviMilica = new string[] { kursevi[4].ime };
            string[] kurseviDavid = new string[] { kursevi[0].ime,
kursevi[1].ime };

            //string[] kurseviAna = new string[] { "python1", "pygame1",
"htmlcss1" };
            Ucenik ana = new Ucenik("Ana", "Savicevic", 2008, false,
kurseviAna);
            Program.uceniciUAlg.Add(ana);

            //string[] kurseviAndjela = new string[] { "python1", "python2",
"pygame1", "htmlcss1" };
            Ucenik andjela = new Ucenik("Andjela", "Stankov", 2008, true,
kurseviAndjela);
            Program.uceniciUAlg.Add(andjela);

            //string[] kurseviMilica = new string[] { "ScratchJR" };
            Ucenik milica = new Ucenik("Milica", "Rakonjac", 2013, true,
kurseviMilica);
            Program.uceniciUAlg.Add(milica);

           // string[] kurseviDavid = new string[] { "python1", "python2" };
            Ucenik david = new Ucenik("David", "Milosavljevic", 2006, true,
kurseviDavid);
            Program.uceniciUAlg.Add(david);
        }

        public static void PopuniListuNastavnika()
        {
            // ovaj dio dodati kad se napravi dodatna klasa za kurseve
            string[] kurseviTanja = new string[] { kursevi[4].ime,
kursevi[5].ime };
            string[] kurseviMarija = new string[] { kursevi[0].ime,
kursevi[1].ime, kursevi[2].ime, kursevi[3].ime };

            //string[] kurseviTanja = new string[] { "ScratchJR", "Tynker",
"HTML CSS JR", "Logo" };
            Nastavnik tanja = new Nastavnik("Tatjana", "Jaksic", 1986,
kurseviTanja);
            Program.nastavniciUAlg.Add(tanja);

            //string[] kurseviMarija = new string[] { "Scratch1", "Python1",
"Python2", "Pygame1", "C#" };
            Nastavnik marija = new Nastavnik("Marija", "Mitrovic", 1986,
kurseviMarija);
            Program.nastavniciUAlg.Add(marija);
        }

        public static void Ispisi(List<Ucenik> ucenici)
        {
Console.WriteLine("++++++++++++++++++++++++++++++++++++++++++++++++");
```

```
            Console.WriteLine("U C E N I C I");
Console.WriteLine("++++++++++++++++++++++++++++++++++++++++++++++++++");
            foreach (Ucenik element in ucenici)
            {
                element.Prikaz();
            }
        }

        public static void Ispisi(List<Nastavnik> nastavnici)
        {
Console.WriteLine("++++++++++++++++++++++++++++++++++++++++++++++++++");
            Console.WriteLine("N A S T A V N I C I");
Console.WriteLine("++++++++++++++++++++++++++++++++++++++++++++++++++");
            foreach (Nastavnik element in nastavnici)
            {
                element.Prikaz();
            }
        }

        // dve funkcije ispisi - isto se zovu ali imaju razlicite argumente.
        // ovaj sistem kodiranja u C# je poznat kao overloading
        // funkcije mogu da imaju isto ime ali da se razlikuju po tipu
podatka ili po broju argumenata
        static void Main(string[] args)
        {
            PopuniListuKurseva();
            PopuniListuUcenika();
            PopuniListuNastavnika();
            Ispisi(Program.uceniciUAlg);
            Ispisi(Program.nastavniciUAlg);
        }
    }

    public class Osoba
    {
        // da bi se videli u nasledjenoj klasi, moraju da budu protected
        protected string ime;
        protected string prezime;
        protected int godiste;

        public Osoba(string _ime, string _prezime, int _godiste)
        {
            this.ime = _ime;
            this.prezime = _prezime;
            this.godiste = _godiste;
        }

        public Osoba()  // ovaj podrazumevani konstruktor mora da postoji
kada imamo nasledjivanje
        {   }

        ~ Osoba()
        {   }
    }

    public class Nastavnik : Osoba
    {
        private string[] kursevi;

        public Nastavnik(string _ime, string _prezime, int _godiste, string[]
_kursevi)
```

```
        {
            this.ime = _ime;
            this.prezime = _prezime;
            this.godiste = _godiste;
            this.kursevi = _kursevi;
        }

        public void Prikaz()
        {
            string tekst = $"{this.ime,15:##} {this.prezime,15:##}
{this.godiste,5:##}";
            Console.WriteLine(tekst);
        }
    }

    public class Ucenik : Osoba
    {
        private bool aktivan;
        private string[] kursevi;
        private string trenutniKurs;

        public Ucenik(string _ime, string _prezime, int _godiste, bool
_aktivan, string[] _kursevi)
        {
            this.ime = _ime;
            this.prezime = _prezime;
            this.godiste = _godiste;
            this.aktivan = _aktivan;
            this.kursevi = _kursevi;

            if (aktivan)
            {
                this.trenutniKurs = _kursevi[0];
            } else
            {
                this.trenutniKurs = "";
            }
        }

        public void Prikaz()
        {
            string tekst = $"{this.ime,15:##} {this.prezime,15:##}
{this.godiste, 5:##}";
            Console.WriteLine(tekst);
        }
    }

    public class Kurs
    {
        internal string ime;
        internal int brojCasova;

        public Kurs (string _ime, int _brCasova)
        {
            ime = _ime;
            brojCasova = _brCasova;
        }
    }
}
```

NAPOMENA 1: Fokusirati se samo na neophodna polja i informacije jer je zadatak veliki

NAPOMENA 2: Protected u Baznoj klasi, a u Nasledjenoj private
NAPOMENA 3: Internal jer su u istom fajlu. On se konta i kao private.